Zahlenumwandlung – Basissysteme

Ac 2013-2021

Es soll gezeigt werden, wie Zahlen (auch **mit Nachkommaanteil , jedoch <u>keine negativen</u> Zahlen**) von einem Basissystem g in ein anderes Basissystem umgewandelt werden können (z.B. 10 → 16).
Dabei werden **Basen von 2 bis 36** verwendet.
Für jede Basis g sind Ziffern von 0 bis g-1 erlaubt !
Bei Basen größer als 10 verwendet man für Ziffern oberhalb von 9 die Buchstaben A(=10), B(=11), … , Z(=35) .

<u>Beispiele für solche Basen:</u>

g = 2 (Dualsystem)	2 Ziffern 0 und 1 ,
g = 10 (Dezimalsystem)	10 Ziffern 0 bis 9 ,
g = 16 (Hexadezimalsystem)	16 Ziffern 0 bis F ,
g = 36 (36-System)	36 Ziffern 0 bis Z .

Die Basis g ist eine natürliche Zahl ≥ 2 .

Jede Zahl z in dieser Basis g mit **n+1 Vorkommastellen** und **m Nachkommastellen** lässt sich eindeutig folgendermaßen darstellen (<u>g-adische Entwicklung von z) :</u>

$$z = \underbrace{a_n g^n + a_{n-1} g^{n-1} + \ldots + a_1 g^1 + a_0}_{\text{Vorkommastellen}} + \underbrace{a_{-1} g^{-1} + a_{-2} g^{-2} + \ldots + a_{-m} g^{-m}}_{\text{Nachkommastellen}} \quad ; a_n \neq 0$$

Kurzschreibweise: $z = a_n\, a_{n-1} \ldots a_1\, a_0\, ,\, a_{-1}\, a_{-2} \ldots a_{-m}$

<u>Beispiele:</u>

$4723{,}58_{10} = 4\cdot10^3 + 7\cdot10^2 + 2\cdot10^1 + 3 + 5\cdot10^{-1} + 8\cdot10^{-2}$

$D7C3{,}A6_{16} = 13\cdot16^3 + 7\cdot16^2 + 12\cdot16^1 + 3 + 10\cdot16^{-1} + 6\cdot16^{-2}$

$110100{,}01_2 = 1\cdot2^5 + 1\cdot2^4 + 0\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 0 + 0\cdot2^{-1} + 1\cdot2^{-2}$

$0{,}05_8 = 0 + 0\cdot8^{-1} + 5\cdot8^{-2}$

$K4Z{,}0T_{36} = 20\cdot36^2 + 4\cdot36^1 + 35 + 0\cdot36^{-1} + 28\cdot36^{-2}$

Die tiefgestellte Zahl gibt jeweils die Basis g der Zahl an .

<u>I. Umwandlung von einem beliebigen g-System in das 10-System (g → 10):</u>

Gegeben ist eine Zahl z im g-System . Gesucht ist das Äquivalent im 10-System.
Die Darstellung im 10-System entspricht der obigen Darstellung im g-System, aber die Anzahl der Vorkommastellen und die Anzahl der Nachkommastellen verändern sich gegebenenfalls .

Beispiel für eine Zahl z im Zweiersystem (g = 2), die in das 10-System umgewandelt wird:

$$z = 1001{,}01_2 = 1\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 1 + 0\cdot2^{-1} + 1\cdot2^{-2} = 8 + 0 + 0 + 1 + 0 + 0{,}25 = 9{,}25_{10}$$

Die gesuchte Darstellung der Zahl z im 10-System kann mithilfe des sog. *HORNER-Schemas* (geschicktes Ausklammern !) berechnet werden.
Die Verwendung des Schemas erkennt man in den Schleifen des weiter unten notierten Algorithmus.

(1) Der Vorkommateil von z sei **zVk** : (Hinweis: zVk ist eine <u>natürliche Zahl</u>)
$zVk = ((a_n\cdot g + a_{n-1})\cdot g + a_{n-2})\cdot g + \ldots + a_1)\cdot g + a_0$

(2) Der Nachkommateil von z sei **zNk** : (Hinweis: zNk ist eine <u>rationale Zahl</u> der Form 0, …)
$zNk = (a_{-1}\cdot g^{m-1} + a_{-2}\cdot g^{m-2} + \ldots + a_{-m}) / g^m$ und somit
$\mathbf{zNk = (\ldots (a_{-1}\cdot g + a_{-2})\cdot g + \ldots + a_{-m+1})\cdot g + a_{-m}) / g^m}$

*Daraus ergibt sich der **Algorithmus** für die Umwandlung $g \rightarrow 10$:*

Gegeben sind:
- Basis g
- Vorkommaziffern a[n] , a[n-1] , ... , a[1] , a[0]
- Nachkommaziffern b[1] , b[2] , ... , b[m-1] , b[m] // b[i] = a_{-1}[i] !!

Berechnung des Vorkommateils zVk :

```
zVk = a[n]
Für i von n-1 ab bis 0 wiederhole
    zVk = g·zVk + a[i]
```

Berechnung des Nachkommateils zNk = 0, ... :

```
zNk = b[1]                 // natürliche Zahl;  entspricht a-1
Für i von 2 bis m wiederhole
    zNk = zNk·g + b[i]     // zNk = nat. Zahlen;  Überlauf bei 'int' oder 'long' möglich
zNk = zNk / g^m            // Division mit Nachkommaanteil; evtl. Periode, dann zNk ungenau
```

Ergebnis:

```
zVk + zNk                  // gewünschte Dezimalzahl
```

Beispielrechnungen :

1) $8A,F5_{16}$, also a1=8 a0=10 b1=15 b2=5

```
zVk = 8
zVk = 16·8+10 = 138

zNk = 15
zNk = (15·16+5) = 245
zNk = 245/16² = 0,95703125
```

Ergebnis also: $8A,F5_{16} = 138,95703125_{10}$

2) $AFFE_{16}$, also a3=10 a2=15 a1=15 a0=14 (kein b)

```
zVk = 10
zVk = 16·10+15 = 175
zVk = 16·175+15 = 2815
zVk = 16·2815+14 = 45054
```

Ergebnis also: $AFFE_{16} = 45054_{10}$

3) $10,1011_2$, also a1=1 a0=0 b1=1 b2=0 b3=1 b4=1

```
zVk = 1
zVk = 2·1+0 = 2

zNk = 1
zNk = (1·2+0) = 2
zNk = (2·2+1) = 5
zNk = (5·2+1) = 11
zNk = 11/2⁴ = 0,6875
```

Ergebnis also: $10,1011_2 = 2,6875_{10}$

4) $HalloKarlo_{36} = 1756423207595580_{10}$

5) $0,201_3 = 0,703703703703\ ..._{10} = 0,p703_{10}$ (Periode !)

II. Umwandlung vom 10-System in ein beliebiges g-System:

z sei im 10-System gegeben (vgl. Formel oben) und soll in das g-System umgewandelt werden:

$$z = a_k 10^k + a_{k-1}10^{k-1} + \dots + a_1 10^1 + a_0 + a_{-1}10^{-1} + a_{-2}10^{-2} + \dots + a_{-j}10^{-j} \;;\; a_k \neq 0$$

Gesucht ist somit die g-System-Darstellung

$$z = c_n g^n + c_{n-1}g^{n-1} + \dots + c_1 g^1 + c_0 + c_{-1}g^{-1} + c_{-2}g^{-2} + \dots + c_{-m}g^{-m} \;;\; c_n \neq 0$$

Der Vorkommateil zVk kann dann eindeutig nach Potenzen von g entwickelt werden, wobei die entstehenden Koeffizienten im Bereich { 0, 1, 2, … g-1 } liegen . Dies geschieht durch fortwährende Abspaltung von Divisionsresten. Man dividiert also durch g und bildet den Rest der Division usw. .

Es gilt: $zVk = c_n g^n + c_{n-1}g^{n-1} + \dots + c_1 g^1 + c_0$

Für die erste Division erhält man $\mathrm{int}(zVk/g) = \mathrm{int}(c_n g^{n-1} + c_{n-1}g^{n-2} + \dots + c_1 + c_0 / g)$
Da aber wegen $c_0 < g$ gilt: $\mathrm{int}(c_0/g) = 0$, bleibt: $\mathrm{int}(zVk/g) = \mathrm{int}(c_n g^{n-1} + c_{n-1}g^{n-2} + \dots + c_1)$.
Daraus folgt: $g \cdot \mathrm{int}(zVk/g) = c_n g^{n-1} + c_{n-1}g^{n-2} + \dots + c_1$
Der Rest ist $r = zVk - g \cdot \mathrm{int}(zVk/g) = c_0$. Somit hat man bereits c_0 abgespalten !
Dieses Abspaltungsverfahren führt man dann für int(zVk/g) statt zVk fort, usw. .

Beispielrechnung für eine Zahl im Zweiersystem (g = 2):

Sei zVk = 37. Es wird nun fortwährend durch 2 dividiert und der Rest notiert:
37 / 2 = 18 Rest **1**.
18 / 2 = 9 Rest **0**.
9 / 2 = 4 Rest **1**.
4 / 2 = 2 Rest **0**.
2 / 2 = 1 Rest **0**.
1 / 2 = *0* Rest **1**.
Beim Ergebnis 0 endet der Algorithmus. Die Reste (**oberster Wert ganz rechts notiert**) sind **100101**.
Dies ist die Dualzahldarstellung von **37**, wie man leicht nachprüfen kann.

Der Nachkommateil zNk lässt sich auf ähnliche Weise bestimmen wie zVk, nur wird hier mit g fortwährend multipliziert .

Es gilt: $zNk = c_{-1}g^{-1} + c_{-2}g^{-2} + \dots + c_{-m}g^{-m}$

Dann erhält man: $\mathrm{int}(g \cdot zNk) = \mathrm{int}(g \cdot (c_{-1}g^{-1} + c_{-2}g^{-2} + \dots + c_{-m}g^{-m})) =$
$\mathrm{int}(c_{-1} + c_{-2}g^{-1} + \dots + c_{-m}g^{-m+1}) = c_{-1}$
Dies gilt, weil alle anderen Teilterme der Klammer im Intervall [0;1[liegen und ihr int-Anteil damit 0 ist !
Wir haben also bereits c_{-1} , den ersten Koeffizienten nach dem Komma, abgespalten !
Der Rest ist $r = g \cdot zNk - \mathrm{int}(g \cdot zNk) = c_{-2}g^{-1} + c_{-3}g^{-2} + \dots + c_{-m}g^{-m+1}$.
Dieser Rest wird im nächsten Schritt wieder für zNk eingesetzt, und das Verfahren wiederholt.
Der Algorithmus endet, falls zNk = 0 wird. Dies ist aber in den meisten Fällen nicht so, weil häufig eine periodische Darstellung für den Nachkommaanteil im g-System entsteht.
Man muss dann eine Schranke für die Anzahl der Stellen angeben.

Beispielrechnung 1 für eine Zahl im Zweiersystem (g = 2):

Sei zNk = 0,25 im 10-System gegeben.
Dann ist int(2·0,25) = int(0,5) = **0** .
Dies ist bereits die erste Stelle nach dem Komma.
Der Rest ist r = 0,5 – 0 = 0,5. Dies ist das neue zNk.
Wir bilden wieder int(2·0,5) = int(1) = **1**. Die zweite Stelle nach dem Komma.
Der Rest ist r = 1 – 1 = 0. Dies ist das neue zNk.
Wegen zNk = 0 bricht hier der Algorithmus ab.

Ergebnis: $0{,}25_{10} = 0{,}01_2$.

Beispielrechnung 2 für eine Zahl im 3-System (g = 3):

```
Sei zNk = 0,75 im 10-System gegeben.
Dann ist  int(3·0,75) = int(2,25) = 2 .  (erste Stelle nach dem Komma)
Der Rest ist  r = 2,25 - 2 = 0,25. Dies ist das neue zNk.
Wir bilden wieder int(3·0,25) = int(0,75) = 0. Die zweite Stelle nach dem Komma.
Der Rest ist  r = 0,75 - 0 = 0,75. Dies ist das neue zNk.
Hier können wir den Algorithmus beenden, denn zNk = 0,75 war bereits zu Beginn
gegeben.
Daher werden sich die beiden ersten Schritte stets wiederholen und damit wird die
Folge 20 immer wieder entstehen. Wir haben eine Periode !
```

Ergebnis: $0{,}75_{10} = 0{,}\overline{20}_3$.

Überlegungen zur Genauigkeit:

Wegen der ungenauen Rechnung mit reellen Zahlen rechnen wir auch für zNk mit natürlichen Zahlen:
Dazu müssen wir nur zNk mit **mult := 10^m** multiplizieren, dann erhalten wir die Ziffernfolge hinter dem Komma.
Da der Exponent von 10 jetzt stets ≥ 0 ist und $c_i \in [0 ; 9]$ treten nun keine gebrochenen Anteile mehr auf .
*Selbstverständlich müssen wir bei der int()- sowie der Rest-Bildung auch wieder durch **mult** dividieren !*

***Algorithmus** für die Umwandlung 10 → g :*

```
Gegeben sind :
 - Basis g                  // Zahl im Bereich 2,3, … ,9,11, … ,36
 - Vorkommaziffern zVk      // Ziffern im Bereich 0, … , 9
 - Nachkommaziffern zNk     // Ziffern im Bereich 0, … , 9
 - maxSt                    // maximale Nachkommastellenzahl, z.B. 200

Berechnung der Koeff. c[k] von zVk :
    i = 0
    wiederhole
        c[i] = zVk mod g
        zVk = zVk div g
        i = i+1
    bis zVk = 0
    für k von i -1 ab bis 0 wiederhole:   notiere c[k]

Berechnung der Koeff. von zNk :
    mult = 10^m                // m ist die Länge von zNk; Überlauf bei 'int' oder 'long' möglich
    zaehler = 0
    wiederhole
        zaehler = zaehler+1
        zNk = zNk·g
        Notiere zNk div mult    // Ergebnisse > 9 in A,B, … umwandeln
        zNk = zNk mod mult
    bis zNk = 0 oder (zaehler = maxSt) oder „Periode erkannt“
```

Beispielrechnung 1 für ein ganzzahliges zNk , umzuwandeln in das 2-System (g = 2):

Sei z = 0,75 im 10-System gegeben, also zNk=75 g=2 mult=10^2 = 100
zNk·g div mult = 75·2 div 100 = 150 div 100 = **1**
zNk = 150 mod 100 = 50
zNk·g div mult = 50·2 div 100 = 100 div 100 = **1**
zNk = 100 mod 100 = 0 // Ende

Ergebnis: $0{,}75_{10} = 0{,}11_2$

<u>Beispielrechnung 2 für ein ganzzahliges zNk , umzuwandeln in das 16-System (g = 16):</u>

Sei z = 0,82 im 10-System gegeben, also zNk=82 g=16 mult = 10^2 = 100
zNk·g div mult = 82·16 div 100 = 1312 div 100 = **<u>13 = D</u>**
zNk = 1312 mod 100 = 12
zNk·g div mult = 12·16 div 100 = 192 div 100 = **<u>1</u>**
zNk = 192 mod 100 = 92
zNk·g div mult = 92·16 div 100 = 1472 div 100 = **<u>14 = E</u>**
zNk = 1472 mod 100 = 72
zNk·g div mult = 72·16 div 100 = 1152 div 100 = **<u>11 = B</u>**
zNk = 1152 mod 100 = 52
zNk·g div mult = 52·16 div 100 = 832 div 100 = **<u>8</u>**
zNk = 832 mod 100 = 32
zNk·g div mult = 32·16 div 100 = 512 div 100 = **<u>5</u>**
zNk = 512 mod 100 = 12 (12 war bereits oben als Rest, also Periode !)
Daher sind die nächsten Ziffern: 1 E B 8 5 1 E B 8 5 1 ...

Ergebnis: $0{,}82_{10} = 0{,}\mathrm{D}\overline{\mathbf{1EB85}}_{16}$

<u>Beispielrechnung 3 für ein ganzzahliges zNk , umzuwandeln in das 2-System (g = 2):</u>

z = $0{,}2_{10}$, also zNk=2 g=2 mult = 10^1 = 10
zNk·g div mult = 2·2 div 10 = 4 div 10 = **<u>0</u>**
zNk = 4 mod 10 = 4
zNk·g div mult = 4·2 div 10 = 8 div 10 = **<u>0</u>**
zNk = 8 mod 10 = 8
zNk·g div mult = 8·2 div 10 = 16 div 10 = **<u>1</u>**
zNk = 16 mod 10 = 6
zNk·g div mult = 6·2 div 10 = 12 div 10 = **<u>1</u>**
zNk = 12 mod 10 = 2
zNk·g div mult = 2·2 div 10 = 4 div 10 = **<u>0</u>**
zNk = 4 mod 10 = 4
(4 war bereits oben als Rest, also Periode !)
Daher sind die nächsten Ziffern: 0 1 1 0 0 1 1 0 0 1 1 0 ...
Ergebnis: $0{,}2_{10} = 0{,}\overline{\mathbf{0011}}_2$

<u>Weitere Beispiele (ohne Nachweis):</u>

$0{,}64_{10} = 0{,}\overline{\mathbf{10100011110101110000}}_2$

$0{,}6_{10} = 0{,}\overline{\mathbf{1210}}_3$

$5919{,}9_{10} = \mathrm{EFJ{,}I}_{20}$

$1/7_{10} = 0{,}\overline{\mathbf{1}}_8$

$6{,}13_{10} = 6{,}\mathrm{4OHA2VOHA2VOHA2VOHA2VOHA2VOHA2}_{36}$

Hinweis zu negativen ganzen Zahlen :

Die einzige Basis, bei der negative ganze Zahlen mit Minuszeichen dargestellt werden, ist die Basis 10.
Alle anderen Basen versehen ihre negativen ganzen Zahlen mit einem **Vorzeichenbit**.

Bei der Basis 2 in der üblichen so genannten **Zweierkomplementdarstellung** ist das Vorzeichenbit eine **1 (negativ)** oder eine **0 (positiv)** an der **höchstwertigen** Stelle.

Für negative Ganzzahlen ergibt sich demnach folgende Darstellung:

- 8bit-Zahl: 1xxxxxxx
- 16bit-Zahl: 1xxxxxxx xxxxxxxx
- 32bit-Zahl: 1xxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
- 64bit-Zahl: 1xxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx

Will man eine negative n-bit-Dualzahl (Bit Nr. 0 bis Bit Nr. n-1) in eine Zahl der Basis 10 umwandeln, so wandelt man zunächst den Anteil mit den Bits Nr. 0 bis Nr. n-2 um und subtrahiert anschließend 2^{n-1} .

Beispiele:

a) Gegeben ist die (negative) 32-bit-Dualzahl

1000	0000	0000	0000	0000	0000	0000	0000

Wie man leicht sieht, ist der Anteil der ersten 31 Bits gleich Null und somit die Dezimaldarstellung auch 0.
Zur Berücksichtigung des Vorzeichenbits 1 muss man aber noch $2^{31} = 2147483648$ subtrahieren .

Ergebnis: Dezimalzahl $z = -2147483648$.

b) Zur Umwandlung einer negativen Zahl von Basis 10 auf Basis 2 geht man umgekehrt vor:
Man addiert zuerst 2^{n-1} , und wandelt dann diese positive Zahl wie gewohnt um.

Gegeben ist die (negative) 8-bit-Zahl -103 .

Erst wird $2^7 = 128$ addiert. Wir erhalten 25.
$25 = 16 + 8 + 1$. 8-bit-Dualdarstellung: 00011001
Da die ursprüngliche Zahl negativ ist, muss das höchstwertige Bit 1 gesetzt werden.

Ergebnis: $-103_{10} = 10011001_2$

Ergänzung: Die Hexadezimaldarstellung ist: $-103_{10} = 99_{16}$